Power Aware Computing Hardware Management Software for Android Mobile Systems

Ari Cotler[1,2], Dean Shi[2], Onur Sahin[2], Ayse Coskun[2]
Glenbrook North High School, 2300 Shermer Road, Northbrook, IL 60062[1]; Boston University Electrical and Computer Engineering Department, 8 St Marys St, Boston, MA 02215, PHO340[2]

Abstract

Hardware resource management software has not been optimized for maximum power efficiency for Android mobile devices, resulting in shorter battery discharging time. We created an Android application that dynamically manages the power usage of the central processing unit (CPU) and the graphics processing unit (GPU). The application allows the user to independently switch on or off each core of the quad core processor as well as to implement CPU and GPU frequency changes from sliders. In order to test the power efficiency of the application, we used a Wattsup power meter which logs power consumption. We tested Antutu, a commonly used CPU benchmark test and Smash Hit, an Android game, with our automatic temperature and power data recording scripts both with the Android system CPU and GPU governors activated as well as with different power profiles. The data shows that it is more efficient to run with a fewer number of active cores and at lower frequencies. However, if less cores are used to split the workload, they are more likely to overheat, leading to performance loss and the need for thermal management. The data also shows that the difference between running cores at medium or high frequency is negligible as the power consumption differed by less than a watt. A new approach to resource and power management for Android mobile devices using custom governors to automate hardware changes can increase power efficiency. The system would be able to efficiently manage hardware usage, reduce power consumption and ultimately increase battery life.

Introduction

- Current Android hardware management software is suboptimal such that Android mobile devices consume more power than necessary, limiting battery runtime.
- Development of custom resource and power management software to manage device power consumption can significantly increase battery runtime.
- When the Central Processing Unit (CPU) or Graphics Processing Unit (GPU) are in high demand, the Android device consumes more power.
- However, not all cores of the CPU need to be active or running at full frequency when a user performs simple tasks such as checking email or using a social media application.
- By dynamically scaling CPU and GPU frequencies and controlling which CPU cores are active at any given time, we are able to increase power efficiency without decreasing functionality.
- We are currently developing a system governor for the Android operating system to increase battery life and lower operating temperature so as to promote maximum hardware efficiency.

Objective

To monitor Android Mobile device hardware usage and increase power efficiency by implementing core and frequency changes.

Methods and Equipment

- Study procedures were performed with a Qualcomm development board with a quad core Snapdragon 600 CPU and an integrated Adreno 320 GPU (Figure 1). A block diagram of the development board is presented in figure 2.

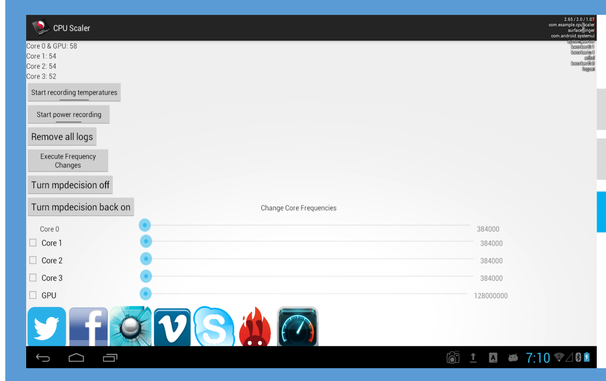

Figure 1. The CPU/GPU scaling application

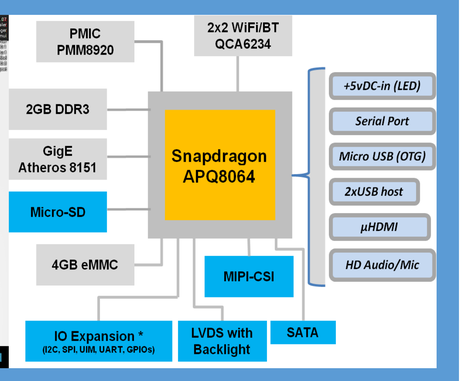

Figure 2. IFC6410 development board block diagram

- Buttons were created to disable the Android operating system default CPU and GPU governors so as to implement custom power profiles (Figure 3).

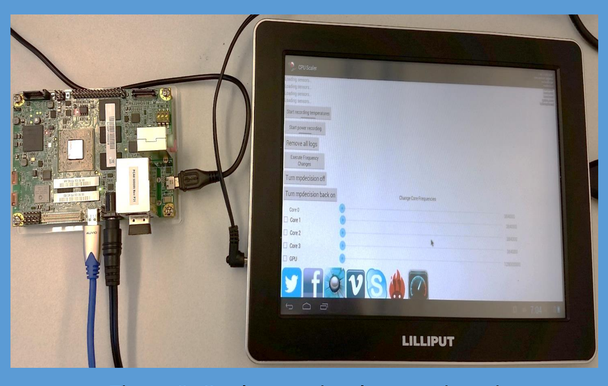

Figure 3. Qualcomm development board

- A Wattsup power meter was used to log data every second over a set interval of time so as to keep a common timescale.
- Two applications were used for testing; Antutu (Figure 4), a commonly used CPU benchmark test and Smash Hit (Figure 5), a CPU and GPU intensive game.
  - A twenty second period of inactivity was used before and after each trial to establish a baseline for trend comparison.
- Shell commands were used to read the system's core and GPU temperature files and send the data to a log file.
- Timestamps were used to track the logging intervals.
- An "if statement" was added to the program to set the four cores to their lowest frequency if the temperature exceeded 90° C, allowing the cores to cool down to prevent the board from crashing.

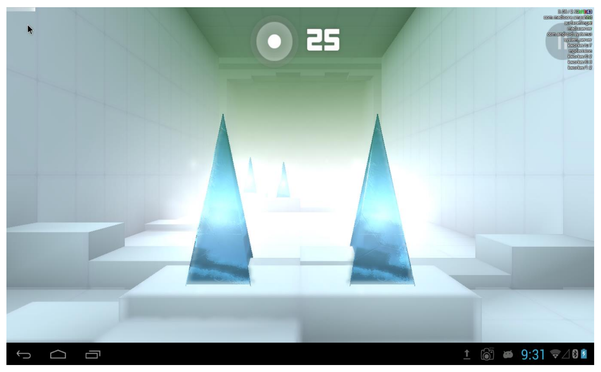

Figure 4. Smash Hit, a CPU and GPU intensive game

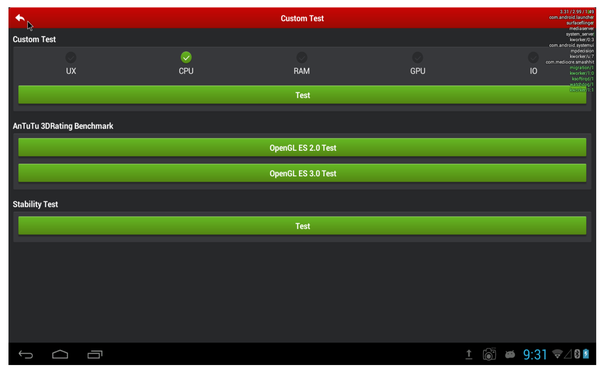

Figure 5. Antutu, a CPU intensive benchmark test

Results

- Power consumption increased as more cores were activated. For each trial, the core(s) running on the lowest frequencies used the least amount of power throughout the duration of the trial (Figure 6).

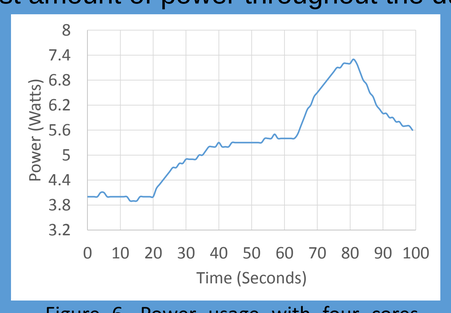

Figure 6. Power usage with four cores (cores 0-3) running Antutu at low frequency

- Power consumption differed only by a fraction of a watt when the CPU was run at medium or high frequency (Figures 7 and 8). In contrast, running the cores at medium or high frequency required two or more watts of power than running the cores at low frequency.

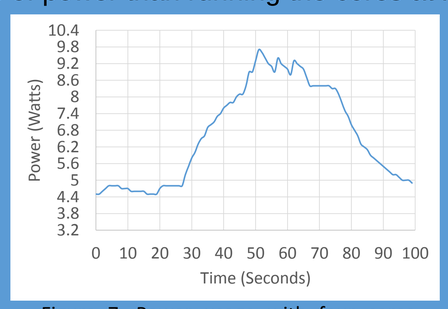

Figure 7. Power usage with four cores (cores 0-3) running Antutu at medium frequency

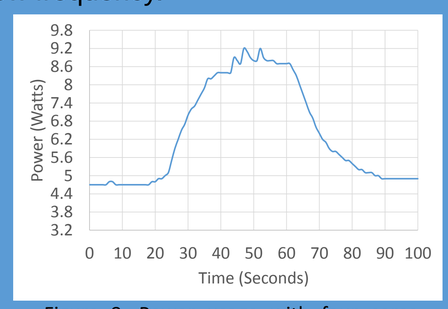

Figure 8. Power usage with four cores (cores 0-3) running Antutu at high frequency

- In a control trial using Antutu with the Android mpdecision CPU governor active, power consumption peaked at 13.2 watts and then decreased (Figure 9). When the governors were disabled and all cores were activated on high, the peak power was only 9.2 watts and remained constant while the application was running, resulting in greater overall consumption (Figure 8).
- Power consumption was compared using three and four cores at high frequencies. When four cores were used, power peaked at 9.2 watts and remained relatively constant (Figure 8). However, when three cores were used, power peaked at 9.3 watts but then rapidly decreased indicating less overall power usage with three cores (Figure 10).

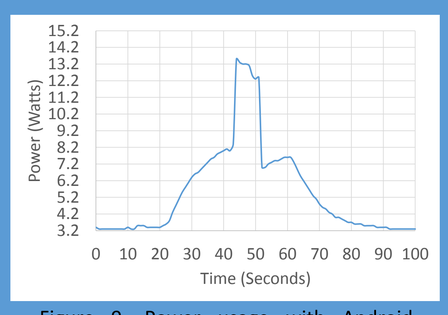

Figure 9. Power usage with Android default governor mpdecision active

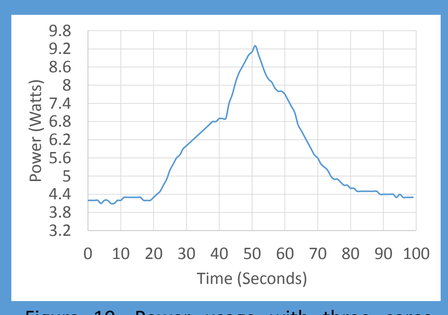

Figure 10. Power usage with three cores (cores 0-2) running Antutu at high frequency

- Core temperatures increased rapidly when the CPU was in high demand and then decreased after the period of utilization (Figures 11, 12 and 13).
- Core temperatures increased more rapidly with three cores running at high frequency (Figure 12) compared to four cores (Figure 13).

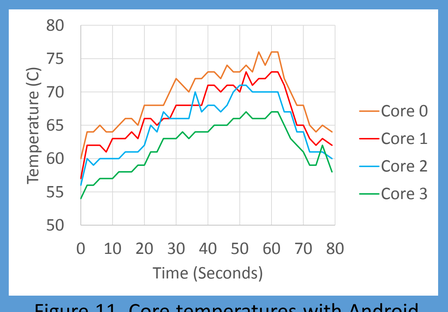

Figure 11. Core temperatures with Android default governor mpdecision active

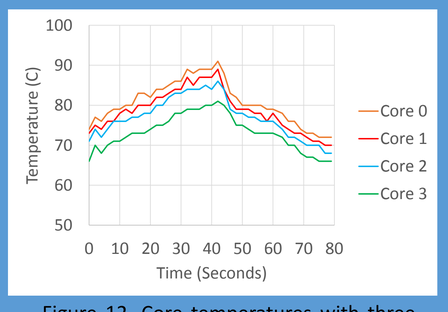

Figure 12. Core temperatures with three cores (cores 0-2) running Smash Hit at high frequency

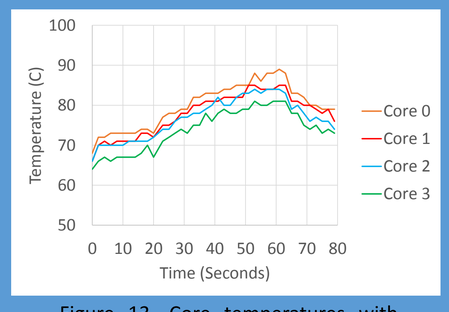

Figure 13. Core temperatures with four cores (cores 0-3) running Smash Hit at high frequency

Conclusion

Power can be conserved when a user is running a non CPU or GPU intensive application by setting the CPU and GPU to lower frequencies and turning off unused cores. However, if less cores are available to spit the workload, the cores will heat up more quickly, which may cause performance loss. Device performance indicators such as frame rate and application runtime will be tested to see how the core and frequency changes affect the user's experience. The next step will be to develop a custom kernel in which hardware and frequency changes occur automatically with custom governors so the user does not need to initiate the core and frequency changes. The long-term goal is to develop an automated system to efficiently manage hardware usage, reduce power consumption and ultimately increase battery life.

References

1. Ayse K. Coskun, "Power Management in Mobile Devices" Circuit Celler 288, 2014
2. Yang, Lei, et al. "HAPPE: Human and Application-Driven Frequency Scaling for Processor Power Efficiency." *Mobile Computing, IEEE Transactions on* 12.8 (2013): 1546-1557.